

**REMARKS**

This paper is in response to the Office Action of April 6, 2004. Upon entry of this Amendment, claims 1-12 and 25-48 will be pending for a total of 36 claims. The due date for response extends to September 6, 2004 with a two-month extension, a request for which is included herewith. Claim 4 stands rejected under 35 U.S.C. §112, second paragraph. Claims 1, 2, 13, 14, 25, and 26 stand rejected under 35 U.S.C. §102(e). Claims 3-12, 15-24, and 27-36 stand rejected under 35 U.S.C. §103(a). Reconsideration of the outstanding rejections is respectfully requested in view of these amendments and Remarks.

***Amendment***

Claims 13-24 are canceled. Claim terminology is modified to be more consistent with terms used in the written description. In addition, claims 37-48 are added. No new matter has been entered.

***Rejection under 35 U.S.C. § 112***

Claim 4 stands rejected under 35 U.S.C. §112, second paragraph for failing to particularly point out and distinctly claim the invention. Specifically, claim 4 depended from itself. Applicant has amended claim 4 so that it now depends from claim 1, thereby obviating the rejection. Applicant therefore respectfully requests withdrawal of the rejection.

***Rejections under 35 U.S.C. § 102(e)***

Claims 1, 2, 13, 14, 25, and 26 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,496,871 issued to Jagannathan et al. ("Jagannathan"). Applicant respectfully traverses for the following reasons: Claims 13 and 14 have been canceled without prejudice, thereby obviating any rejections thereagainst; claims 1, 2, 25, and 26 contain features that distinguish them from Jagannathan.

Jagannathan discloses a runtime agent that allows computer programmers to write active objects that can simultaneously span multiple heterogeneous machines and which can

be migrated in part or in whole from one machine to another. While helpful to computer programmers, this is very different from the invention described and claimed in the instant application in which a computing environment is encapsulated into a compute capsule. Because each object in Jagannathan exists in the object space defined by the agent and each agent operates in the context of one or more bases, specific information about the system environment is not needed. This is along the lines of the virtual machine technique used by Java and related technologies which are discussed in the Background section of Jagannathan.

In addition, while Jagannathan does mention “encapsulation” it does not have the same meaning as “encapsulation” as presently disclosed. Specifically, Jagannathan talks of encapsulation as placing an object in a “protection domain” (see, e.g., col. 8, lines 65-68) provided by an agent. This is different from the meaning of “encapsulation” as presently disclosed, which is directed to a representation of processes and associated state.

Furthermore, unlike the agent described in Jagannathan, compute capsules comprise one or more processes and their associated system environment. The system environment in a capsule comprises state information relating to what the processes are doing at a specified time. System environment information may include, for instance, privileges, configuration settings, working directories and files, assigned resources, open devices, installed software, and internal program state. Thus, the processes need not exist in a predefined object space that exists across multiple machines and is mobile from one machine to another since the system environment in which the process resides is effectively represented within the compute capsule.

Thus, Jagannathan does not provide the advantages exhibited by the present invention, including the possibility of suspending and storing a user session, restarting a user session, and migrating a user session. Jagannathan in contrast allows a computer programmer to design a program that can be migrated or distributed, not a user session or active computing environment. Jagannathan does not permit a user to create capsules with arbitrary contents or entire login sessions.

Claim 1 is amended and now sets forth, inter alia, a method for managing an active computing environment including encapsulating one or more active processes into a compute capsule, and encapsulating a system environment relating to said processes into said compute capsule, said system environment including host-specific data. Support for this amendment can be found, e.g., in the written description at page 11, lines 12-19 and page 15, lines 7-9 of

the substitute specification dated June 4, 2004. Jagannathan teaches only “encapsulation of a computation (i.e. a task) and related data” (col. 4, lines 34-35) and that “Each agent . . . encapsulates a collection of objects, including simple objects (such as data objects) as well as a collection of threads or concurrently executing tasks” (col. 9, lines 1-4). Jagannathan does not teach encapsulating a system environment nor does Jagannathan teach encapsulating host-specific data.

In the outstanding Office Action, the Examiner points to col. 10 lines 47-64 as showing “encapsulating a system environment” (page 3, lines 3-4 of the outstanding action). Applicant respectfully disagrees. Here, Jagannathan is merely showing the distributed nature of the agents. While Jagannathan’s objects may be distributed among a plurality of machines, there is no suggestion that system environment information be encapsulated.

Likewise, the Office Action points to col. 9, lines 13-31 as showing a system environment comprising an associated state of active processes (page 3, lines 7-9 of the outstanding action), but Applicant respectfully disagrees. The term “state” is used by Jagannathan to describe the agent’s current state, not the state of the so-called encapsulated objects and associated data. The agent defines the object space in which the encapsulated objects can exist and protects them. Jagannathan only teaches encapsulating objects and data, not environmental state information as presently disclosed. See Jagannathan, col. 9, lines 1-4.

For the reasons stated above, Applicant respectfully submits that claim 1 is neither anticipated, nor rendered obvious in view of, Jagannathan. Claims 2-12 and 40-44 depend from claim 1 and should be allowed for at least the same reasons as claim 1. Claim 25 contains features similar to those in claim 1 and should be allowed for the same reasons as claim 1. Claims 26-39 depend from claim 25 and should be allowed for at least the same reasons as claim 25.

***Rejections under 35 U.S.C. § 103(a)***

Claims 3-12, 15-24, and 27-36 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Jagannathan in view of the publication by Schrimpf (“Schrimpf”). Applicant respectfully traverses because not all features are taught or suggested by the prior art and because there was no motivation or suggestion to combine and/or modify the references as described in the outstanding rejection.

1. *Not All Limitations Taught or Suggested By Prior Art*

As mentioned above with regard to the rejections under 35 U.S.C. § 102, Jagannathan does not teach “encapsulating system environment information” related to encapsulated processes. Jagannathan teaches instead a programming tool that allows computer programmers to easily create distributed and movable objects using agents to provide a shared memory abstraction in a distributed networked environment (col. 9, lines 24-26). The “state information” referenced in the Office Action (in the bottom paragraph of page 4 of the Office Action) is in reference to the agents which provide the object space in which the distributed objects exist, and this information is not encapsulated.

Schrimpf teaches an operating system with the ability to promote load balancing and sharing among a plurality of processors in a distributed system by migrating existing processes. However, Schrimpf does not mention encapsulating a process with its environment, nor encapsulating a plurality of processes. Thus, neither Jagannathan nor Schrimpf teaches encapsulating a system environment. While Schrimpf does teach copying data related to a process and its state, no mention is made of encapsulation or any analog thereof. Furthermore, Applicant respectfully submits that Schrimpf implicitly teaches away from encapsulation. First, creating a compute capsule representing an active computing environment would not necessarily aid in load balancing if that environment represented a great deal of load on system resources, since the entirety of that load would simply be placed onto another resource and the load would still be unbalanced. Secondly, the encapsulation steps would add to the number of steps and therefore the time it would take to migrate the processes. Schrimpf explicitly teaches that migration should take as little time as possible. See, e.g., page 72, lines 4-5 of the first full paragraph: “Migration . . . should not interrupt execution for long.” Thus, Schrimpf implicitly teaches away from encapsulation as it would have increased the time necessary to make the migration and it would not necessarily have been helpful in load balancing.

In addition, while Jagannathan does mention “encapsulation” it does not have the same meaning as “encapsulation” as presently disclosed. Specifically, Jagannathan talks of encapsulation as placing an object in a “protection domain” (col. 8, lines 65-68). This is different from the meaning of “encapsulation” as presently disclosed, which is directed to a representation of processes and associated state.

Since neither Jagannathan nor Schrimpf teach or suggest encapsulating a system environment such as set forth in claim 1 and similarly in claim 25, and because all pending rejected claims depend from one of these independent claims, each and every feature set forth in the claims is not met by the cited prior art. Applicant therefore respectfully submits that the claims are allowable.

2. No Suggestion or Motivation to Combine and/or Modify References

The Office Action suggests that “it would have been obvious . . . to combine Jagannathan and Schrimpf since the method of Jagannathan, while allowing portability of processes and maintaining state information, fails to specifically disclose what types of state information may be encapsulated within a process.” As mentioned above, Applicant respectfully disagrees that Schrimpf teaches encapsulating system environment information related to encapsulated processes. However, even if, for the sake of argument, Schrimpf did teach such encapsulation, there would have been no motivation to combine with Schrimpf.

Jagannathan teaches an agent and system that allows computer programmers to create distributed objects. Schrimpf teaches a system for migrating processes. While both references deal with distributed software, they are not combinable since Jagannathan allows development of application-level software and Schrimpf is an operating system and therefore operates at a much lower level, i.e., closer to hardware. Jagannathan provides a shared memory abstraction layer for computer program developers while Schrimpf provides a low-level operating system that allows for migrating processes for the purposes of load balancing. These systems are simply incompatible and cannot be combined.

Specifically, the specifics of core image or other state information mentioned by Schrimpf (page 72, first paragraph) are not available to Jagannathan, since Jagannathan is directed to agents or “protection domains” which exist on top of an underlying operating system. Even if Jagannathan had access to such information, it would not be able to utilize it.

Furthermore, Applicant respectfully disagrees that sufficient motivation to combine the references can come from the fact that Jagannathan lacks information supplied by Schrimpf. First, Applicant disagrees that Jagannathan lacks information relating to “what types of state information may be encapsulated within a process” as suggested in the outstanding action. Applicant regards Jagannathan as being quite clear that “each agent 40


encapsulates a collection of objects, including simple objects (such as data objects) as well as a collection of threads or concurrently executing tasks" (col. 9, lines 1-4). Secondly, even if Jagannathan was lacking as the Office Action suggests, this is insufficient motivation to combine with Schrimpf. Merely lacking in the information indicated would not have led a person having ordinary skill in the art to look to Schrimpf since Schrimpf does not relate to mobile agents which is the subject matter of Jagannathan.

Therefore, since there was no suggestion or motivation to modify and/or combine Jagannathan and Schrimpf in the manner set forth in the Office Action, Applicant respectfully submits that the rejections under 35 U.S.C. § 103(a) are improper and should be withdrawn.

Applicant submits that the prior art does not teach or suggest certain aspects of the invention now claimed and respectfully submit that the application is now in condition for allowance. A Notice of Allowance is therefore respectfully requested.

If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 749-6900 x6933. If any other fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP583). A copy of the transmittal is enclosed for this purpose.

Respectfully submitted,  
MARTINE & PENILLA, LLP

  
Leonard Heyman, Esq.  
Reg. No. 40, 418

710 Lakeway Drive, Suite 170  
Sunnyvale, CA 94085  
Telephone: (408) 749-6900  
Facsimile: (408) 749-6901  
**Customer Number 32291**